$ 1.50

denver apple pi SEED

Newsletter for Denver's Apple users

Volume 7 Number 5 May 1985

PRESIDENT'S REPORT

We have come to the end of another year for the officers of Apple Pi. In May, new officers are elected and the old officers drift off to some obscure location, only to be seen occasionally at a general meeting or CORE meet. Before the new officers take the helm, I wish to take this opportunity to thank all the outgoing officers. Andre' vanMeulebrouck, who brought us interesting meetings and "The WOZ;" Liane Nusse, who faithfully transferred all the files to PFS and has worked hard as Secretary and really has a leg up on knowing software for the children. Then there is Will Smith, as Treasurer, I was sure he was going to open the 1st Bank of Apple Pi. Will provided sound guidance and financial planning to allow Apple Pi to upgrade its BBS to a hard disk, 1200 baud system, provide two computers for library copying, plus still maintain enough money to keep the club store running. Randy Ansley served not only as a Director, but as the Editor of the SEED. With Randy, the SEED has become a letter quality document of the highest quality. Delays in getting SEEDs are still a problem with the Postal Service, but Randy kept trying to get it out earlier. The membership now expects to receive their SEED prior to the meeting, little do some of them know that only two years ago we weren't always sure there was going to be a SEED. Bill Hatchett should be remembered for his cookies at the Christmas Party and being the guy always willing to help another member.

Steve Feldman, who was the newsletter distribution officer, was never seen in the fore front. His duties were behind the scenes. Jeff Beehler, new member contact officer, answered questions at the general meetings and normally operated the computer for the librarian. Jeff, a junior at Arvada West, just participated in a team from Arvada West that took Second in State for computer programming. Jeff is a good guy to know and seems to always know the answers to your questions. As he leaves office, remember he is one of those who has answers. Art Vallee started during the middle of the year as store representative. He arranged for the sales by our store sponsors at the booth outside the meeting hall and distributed the club flyers to the various stores around the city.

Roy Dean, sales officer, is probably the most known officer in the club. I know of no one who hasn't bought something from Roy in the past year. He has been outstanding at getting the club items at terrific prices. Where else can you get 10 disks for $11.00. All of Roy's hard work is deeply appreciated by all of us in the club. We also thank his wife, Debbie, for her help in transporting items to the meetings for the sales table. Steve Martin, the BBS "Guru" has kept the system alive and well for about two years now. During this past year Steve has upgraded the software, added new features, installed the 1200 baud capability, and now has the 10MB hard disk up and operating. Due to the heavy use over the past year, Apple Pi had to impose a 15 minute time limit per call during 2 p.m. and 10 p.m. to allow maximum use by BBS users. Joe Nagy, special interest group officer, has got the beginnings of a MAC users group within Apple Pi. He generated their own by-laws disk and has even sponsored training meetings with Adams Computers for members.

Last, but not least, I wish to thank the Librarian, Trent Hein, who has given generously of his time to do the librarian's duties faithfully each month. These duties require making the monthly DOM for the membership, about 100 copies for sales, demonstrating the DOM at the general meeting, providing members copying from the lebrary at all the CORE meets, making and sending out the Newcomer's disk and library disk, and handling coorespondence with other clubs. Trent also has been valuable in assisting me in answering calls on the Hot Line during the weeks.

To all the officers who have helped and served, I applaud you personally and on behalf of the club. Without your help, the club wouldn't be the prestigious organization it is today. I trust you have each received something in return for your work in the way of a thank you from a satisfied member. I will never forget our year together.

Bob Hein, President

$2

denver apple pi

+:+:+:

$3

The LISP Environment

by André van Meulebrouck

When looking at languages, it is interesting to look at their grammatical makeup (in the case of natural languages), or the underlying basis for the language (in the case of synthetic languages).

The old adage "nothing comes for free" is perhaps as worn out as it is inescapably true. In languages, for example, prices must often be payed for the grammatical makeup of a language. This is not to say anything about equality however. While tradeoffs are a fact of life, they are not by any means equal. Some are VERY expensive.

Since everyone knows a natural language, but not everyone knows a synthetic one, my example of a tradeoff will be from two natural languages.

In English, the order of words is more or less fixed. For example, "the dog ate the cat" means something different when the word order is switched: "the cat ate the dog". Word order in English shows the grammatical roles the words play. A subject must precede its verb, and the object must follow the verb. If the verb and object places were interchanged, the object would become the subject, and vice versa.

In Russian (and probably Latin, since Russian grammar was taken from Latin), this is not the case. Russians can change word order without causing confusion. This is because Russian words have their grammatical information appended (or "coded") on to the end of them. This makes them free to move, since the grammatical information is part of each word rather than being expressed through word order.

Russian gains some expressive power by being able to move word order (to create subtle nuances for example). It also gains some power in long and complex sentences by this type of grammatical makeup. For example: Subject, clause 1, clause 2, clause 3, clause 4. In English, by the time clause 4 goes by, the listener may have lost track of what the subject of it was. In Russian, the grammatical information appended to the words of the clauses help to clear up alot of confusion about what modifies what, and, what is the subject of which clause.

The idea that some cultures solve grammatical problems in such radically different ways from us is intriguing. Most English speakers never exposed to these ideas probably would never imagine or devise such a system if they were inventing a new system! However, the Russian system does have some problems.

What happens if one word requires contradictory grammatical information by virtue of playing dual roles?

The sentence: "I saw him leave" is really two sentences; "I saw him", and, "he left". So, the "him" in the original sentence is the object of "saw" and the subject of "leave"!

I'm not going to go into how Russian resolves this. That's beyond my point. What is of interest here is the analogy to synthetic (computer) languages. There will always be problems that arise, and the question is; what solutions are available, which solutions are eloquent, and, which solutions in themselves will cause other problems that will also have to be resolved.

Now, an example of a radically different approach to a problem in synthetic languages.

The problem will be: how to pass parameters from one routine to another.

LISP's solution to the problem is very different from other languages, and, like the example from natural language, may prove to be a solution that would never dawn on a lot of people.

Basic's solution is to make everything global. Nothing gets passed, everything is known to every routine. This means, if you add a routine to someone else's basic program, you should be very careful how you pick variable names so you don't side affect variables in the original program. If you have a for loop that uses i in your routine, and the original program had an i, you may be inadvertently changing i for the entire program!

Another approach is to have one routine pass a pointer to a variable to another routine. This gives both routines access to the variable, but at the same time, it is a little more exclusive knowledge. In other words, no routine gets the power to change a variable, except by "special invitation" (getting a pointer to it). Fortran uses this method. This method is known as: "call by reference" because the variable is referenced via pointers (the caller has one pointer, the called routine has another pointer, but they both reference the same variable).

Yet another method is to have one routine pass a

LISP

copy of a variable's value, but not the knowledge of the actual location of it. This means the called routine will know what value the variable has, but not where it resides in memory. The called routine can change the copy it receives but it cannot change the value of it for the caller. This type of parameter passage is known as: "call by value". Call by value, when employed, is almost always chosen for safety purposes, but passing copies of variables such as arrays can get expensive in terms of time and memory. Also, sometimes you WANT a called routine to be able to side affect the original variable.

Pascal and C allow both call by value, and call by reference for parameter passing.

LISP's solution to parameter passing is most unusual, and reflects a very different way of looking at the problem. LISP maintains a list called the environment list. This list is globally available, however, the values in it are ordered in a temporal way which affects the access of them. The environment is added to on the left (a binding), and purged from left to right (an unbinding). In that sense, it is like a stack, which is a first in, last out structure. Unlike a stack however, some values imbedded in the middle of it can be accessed and purged.

Here is how it works. When a routine calls another routine, the caller's data is bound with the variable name it corresponds to in the called routine. This "binding" is then placed in the leftmost position of the environment. When a called routine completes its evaluation, an unbinding occurs. This is done by searching the environment from left to right to find the first occurrence of the variable name, then it is purged. Needless to say, the environment is going to be constantly and dynamically changing.

An example will help to show how uniquely dynamic this approach is. The following examples are two LISP functions; test1, and, test2. Test1 calls test2 with the first element of the list that test1 was called with from the keyboard. Test2 prints out variable x (which is a variable defined in test1). Then, test2 prints out its variable: y. The following is a session done on the PRIME computer at UCD via telecommunication, but it could also be done in XLISP (from our library) in a similar fashion.

```
slist example      /* definitions of the test
                      functions */
(defun test1 (x)
(test2 (car x))
)
(defun test2 (y)
(print x)
y
)
OK, lisp example   /* invoke the interpreter with the
                      example functions */

test1              /* interpreter notes that test1 is
                      now in the environment */
test2              /* interpreter notes that test2 is
                      now in the environment */
test1              /* request for the interpreter to
                      show what's bound with test1 */
(lambda (x) (test2 (car x)))

test2              /* request to show test2
                      definition */
(lambda (y) (print x) y)

(test1 '(a b c))   /* test1 invoked with argument */
(a b c)            /* result of printing x       */
a                  /* value bound with y         */

q                  /* exit to system             */
```

Representing the environment as a list with sublists could help clarify what is happening. This list in my examples that follow will be non-standard in the sense that lists in LISP must start with a function name, and the rest of the list is interpreted as arguments. Each sublist of my example list will have a variable name, and the value bound with it rather than a function name followed by arguments.

Before test1 is invoked the environment looks like this:

```
((test2 ..definition..)(test1 ...definition...))
```

After test1 is invoked:

```
((x '(a b c))(test2 ..stuff..)(test1 ..stuff..))
```

After test2 is invoked from test1:

```
((y 'a) (x '(a b c)) (test2 stuff) (test1 stuff))
```

Test2 prints x, then shows the value bound with y, then it returns control to test1. Y becomes unbound:

```
((x '(a b c)) (test2 stuff) (test1 stuff))
```

Test1 returns control and x is unbound:

((test2 stuff) (test1 stuff))

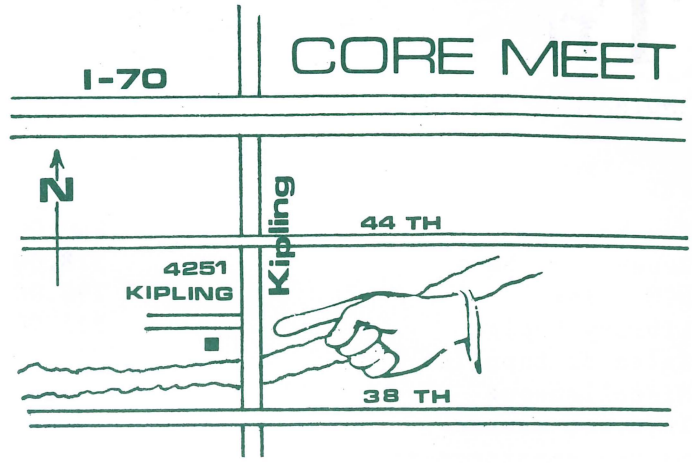
The "y" in test2 after the "print x" could have been "print y" instead, but for cosmetic reasons this was not done. The print statement prints something, and in some cases, returns a value also, which might result in y's value being printed twice.

What should be noted in the example, is that test2 printed out a variable it was not passed and did not own. It was able to do this only because test1 had not completed evaluation, thus, "x" which is test1's variable still had a binding in the environment. When a function completes evaluation, the leftmost bindings that correspond to the names of its variables become unbound. This means that the x printed out by test2 was whatever happened to be bound in the environment at that time. In the context of a different call chain, x could have been completely absent or completely different, which would of course change what test2 would have found when it searched the environment for an x binding. If there had been numerous unresolved recursive calls, involving a function with a variable (parameter) called x, the environment would reflect this by having a binding for each recursive call. Thus, test2 would find the latest binding when searching for an x, which would be the leftmost binding of x in the environment.

Note also, that the environment can be side affected before a call chain is set in motion. This also can affect the outcome of evaluations of a call chain. Test1 and test2 were present in the environment before the call chain was set in motion.

Lastly, note that because the call chain depends on the environment, and the environment is so dynamic, it is impossible to talk about parameter passing apart from the environment that was in effect at the time the call chain was set in motion. With functions that side affect the environment in more radical and direct ways, the status of variables in a call chain could become very muddled. Such functions like setq side affect variable bindings directly (like an assignment statement) and should be avoided when recursing. Examples of setq may be shown in a future article.

As to the question: "Is this an eloquent solution to the parameter passing problem?", the reader is left to decide.



KIDD STUFF

by Lianne Nusse

MAY APPLE PI DEMO--

Members - please let us know what you would like to see demonstrated, in order
that we may reach a more varied interest area. In May we will demonstrate a trivia game with four controllers, which is not only great fun - but very educational as well! The Apple pi demonstrations are always at the CORE meet, from 10 a.m. to 12 a.m. Hope to see you there!

KID'S STUFF - APPLESEEDS
Review - Spinnaker's Grandma's House
Mouse or joystick needed.
Educational value; computer function familiarity, eye-hand coordination.
This program is simply a computerized dollhouse! The little person is presented with the inside of Grandma's empty house and can pick characters to live there - the characters may then travel to various "faraway places" (i.e., circus, town, appliance place, furniture place, beach, etc.) and pick up items to take back to Grandma's house and put anywhere they desire. In our day care environment, it is very interesting to watch the difference between boys and girls operating this software. In general little girls are very precise and have an "a place for everything, everything in it's place" attitude - on the other hand little boys have a tendency to be more "creative", placing things like street lights on walls, and even an occasional item (a refrigerator perhaps?) on Grandma's head!!! At any rate it is an excellant "fun" program, every child here loves it!

Tres. $

ENDING CASH BALANCE	2-28-85	5533.54
INCOME		
New Members		840.00
Renewal Members		378.00
DOM Sales		190.00
Library Copies		221.00
Sales of Supplies		2018.00
Miscellaneous		1.75
Interest Earned		20.93
TOTAL INCOME		3669.68
EXPENSE		
Supplies for Resale		4237.69
Hot Line		92.00
CSM Room Rent		0.00
Prin* g Costs		415.00
Postage		76.48
Supplies & Copies		12.73
BBS Expense		0.00
State Sales Tax		0.00
Subscriptions		47.85
Club Socials		30.00
TOTAL EXPENSE		4911.75
ENDING CASH BALANCE	3-31-85	4291.47
ASSETS	3-31-85	
Cash on Hand		25.00
Cash in Bank		4291.47
Inventory for Resale		5406.19
Prepaid Expense		164.47
Accounts Receivable		0.00
Equipment		0.00
Total Assets		9887.13
LIABILITIES		
Sales Tax Accrued		326.59
Accounts Payable		450.00
Unposted Income		0.00
Total Liabilities		776.59
NET WORTH		9110.54

THE STORE MEMBERS

COLORADO SOFTWARE CONNECTION
P.O. BOX 29869 DENVER, CO 80229-0869 (303) 756-8819

Connecting Point

adam's computers, inc.
3355 So. Wadsworth Blvd.
Lakewood, CO 80227 (303) 986-9655

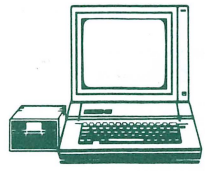
colorado Computer Systems, Inc.
3005 West 74th Avenue
Westminster, CO 80030 (303) 426-5880

ComputerLand®
8749 Wadsworth Blvd.
Arvada, CO 80003 (303) 420-1877

ComputerLand®
2422 So. Colorado Blvd.
Denver, CO 80222 (303) 759-9140

The Neighborhood COMPUTER STORE
13045 West Alameda Parkway
Lakewood, CO 80228 (303) 988-9140

JOHNNY APPLESEED



Dear John,

I am not a member of Apple Pi, I don't have a computer yet. I can't afford to pay for a new Apple but was wondering how to tell if a used Apple was any good. Is there a way to tell if one is on the way out (so to speak)? Do older computers break down a lot? Any help you cgveis appreciated.

LOOKING

Dear LOOKING,

Most of the time you will find that a used Apple will provide great service. You can test a computer just by running it. If it works, it works! The computer itself has very little that can go bad. A chip may need replacing from time to time but that is rare. What has the most chance of going out on you are the things that move. Keyboards, power switches, disk dives and so on. If you are looking at a used computer, it would be wise to pay special attention to those items.

John

Dear John,

I have seen some programs that RUN when you press the RESET key. Is this something that is hard to do? I would like to keep people from stopping my programs with RESET.

RESETLESS

Dear RESETLESS,

The easiest way to stop the normal use of RESET is to change one byte in memory. This zero page location will make the program in memory RUN if the RESET key is pressed. The location is $D6 (214 in decimal)and to go to RUN mode put a $80 (128 in decimal) at that location. If you are programming in BASIC just type POKE 214,128. To go back to normal type POKE 214,0.

John

Dear John,

This is going to sound dumb but what is a compiler?

????

Dear ????,

A compiler is a program that will change a program you write into a program that the computer finds easy to run. In other words you can write a program in BASIC and it will run (we hope). When you run it as a BASIC program each command is sent to a BASIC interpreter to see what is to be done. This is the main reason that BASIC programs run slow. A compiled program has had all the commands interpreted and changed into code that will run without the interpreter so the programs run much faster. Compiled programs also take up more memory so you do have a little trade off.

John

Send your questions to:

Johnny Appleseed
% Denver Apple Pi
P.O.Box 17467
Denver, CO 80217-0467

denver apple pi

DIRECTORY

04-05-85 DIRECTORY PAGE 1

LAST NAME	FIRST	ADDRESS	CITY	ST	ZIP	PHONE	BUSINESS	DATA COMM	EDUCATION	GAMES	MATH / SCIENCE	CP/M	BUSINESS	FORTH	MACHINE LANG	PASCAL	TELECOMM	UTILITIES	WORD PROC	MISC	LANGUAGE	ADVENTURE	GRAPHICS	EDUCATION	INTEGER	PRINTER	NEWSLETTER / DISK	RHODES	OUE	MNL
Anderson	Russell	2595 S Cimarron	Aurora	Co	80014	671-0960				1	1	2			1	1	1						2				n	y	y
Angelos	Tom		Littleton			798-5435																					y	n	n
Aronson	H Harold	5041 W Fair Ave	Littleton	Co	80123	794-7936	1	1			1					1		1	1						1			n	n	
Bobb	Gary	4517 W Ponds Circle	Littleton	CO	80123	798-6566	1	1			1	1					1	1												
Bohn	Frank C	7313 W 76th Ave	Arvada	CO	80003	422-4296		1			1	1				1		1			1	1	1	1						
Bridges	Joe E	867 S Colorado Blvd	Denver	Co	80222	779-9884	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	n	y	y
Campbell	Wallace	3030 Galena Way	Boulder	CO	80303	499-3270		1	1	1	1		1					1	1						1	1	1	n	n	n
Dunlap	Joyce	918 S Pennsylvania	Denver	Co	80209			1	1		1	1	1			1	1	1							1	1		y	y	
Eisler	David	2567 E Geddes Pl	Littleton	Co	80122	694-6217			1	1			2		2			1					2	1	2					
Fales	Clifford	1435 Urban Way	Lakewood	Co	80228	758-9040		1				1					1	1	1				1		1	1				
Forman	Marsie		Wheat Ridge				1	1	1	1	1		1				1	1	1				1		1	1				
Haycraft	Tom	11601 W 77th Dr	Arvada	Co	80005	424-8517	1	1					1		1		1	1	1							1				
Hein	Trent		Arvada					2			1		2		2	2	1	1								2	+		
Hendrix	Michael	4229 S Cherokee St	Englewood	CO	80110	789-2665					1		1	1	1	1			1				1	1			n	n	y	
Justice	Dick	786 S 9th Ave	Brighton	Co	80601	659-0686			2		1		2																	
Kellum	Don	1045 Miami Way	Boulder	Co	80303	494-1314	1		1	1			1													1				
Knoll	Lee	11682 W 36th Ave	Wheat Ridge	Co	80033	421-3706	1	1		1			1			1			1							1				
Kreutzer	John L	1214 Fraser	Aurora	CO	80021	341-2727			1		1		1				1	1										n	n	n
Krieg	Bill	1637 S Oakland Ct	Aurora	Co	80012	750-6952		1	2		2		2					1	2									n	n	n
LeVine	David S	3001 Oak	Lakewood	Co	80215	238-5661	1	2	1		1	1	1		1	1		1	1								1	n	n	y
Mantione	Joe	690 Quince Circle	Boulder	CO	80302	440-9522	1	1		1			1				1	1	1							1	1			
Mayfield	Don	1935 Dudley	Lakewood	Co	80215	233-3174			1				1				1		1									n	y	y
Mehlbach	Kim	11925 W 61st Ave	Arvada	Co	80004	421-0682	1		1																	1		n	n	y
Miller	Jay D	7820 W Paine Ave	Lakewood	CO	80235	986-6659	1	1	1		1						1		1	2						1		n	n	n
Murray	Ronald D	6261 W Arbor Ave	Littleton	CO	80123	797-8980	1	1				1						1	1							2		n	y	
Nusse	Liane	708 Cheyenne	Golden	CO	80401	279-5763		1	1																					
Olmstead	Robert		Golden	Co																			1					n	n	n
Pott	Kathy	10762 W 35th PL	Wheat Ridge	CO	80033	425-8925			1	1	1																			y
Reagan	"3"	6943 W 71st Pl	Arvada	CO	80003	424-2686	2	2	1	2	2	1	2		1	2	1	2	2							1				
Schiffmacher	Edward R	2155 Emerald Rd	Boulder	CO	80302	443-2766	1	1				1	1		1		1		1							1	1	n	n	n
Schucker	Russell	18929 W 60th Pl	Golden	Co	80401	279-7633								1												1				
Shaw	Les	995 McIntire	Boulder	Co	80303	499-0946	1	1	1		1		1				1	1	1							1	1	y	n	y
Stoffel	John L	4181 S Narcissus Way	Denver	Co	80237	757-6302			1	1							1		1				1					y	y	
Thompson	Edwin D	11768 E Atlantic Pl	Aurora	Co	80014	696-7620	1			1			1	1				2							2		1	n	y	y
Thompson	James E	13550 W Ohio Ave	Lakewood	Co	80228	987-1086	1	1	1								1		1											y
Tracy	Wm B	1616 Glenarm Pl #200	Denver	Co	80202	595-8585	1												1			1						n	n	n
Webb	Lee	7306 Mt Sherman Rd	Longmont	CO	80501	530-1185			1		2		1		1		1	1	1					1	2	1		n	n	
Willett	George	3485 Miller	Wheat Ridge	Co	80033	233-5824			2		2	1	1	1	1	2	2	1	1									n	n	n

PASCAL

by Tom Schulz

UCSD PASCAL TURTLEGBAPHICS
A SUMMERY OF COMMANDS TEXT VS GRAPHS

Two kinds of display are available in UCSD Pascal (not on standard Pascal):

1. TEXT mode - normal screen used for, written input/output.
2. GRAPh mode - separate mode for line drawings.

NOTE: Drawings cannot be made in text-mode nor can WRITE & WRITELN be used in graph mode.

TO USE GRAPHICS

1. Immediately after the program heading put USES TURTLEGRAPHICS; This calls the graphics package in the UCSD Pascal library.

2. In your grogram (usually in the main body) put INITTURTLE; This is a built-in procedure that clears scree, changes from text mode to graph mode, and places the "turtle" in the screen's center facing dur right. (i.e. this sets up Pascal to work in graph mode). PENCOLOR is set to none.

INITTURTLE need only be called once and usually in the first statement of your grogram.

3. To return to normal text mode from gph mode us TEXTMODE; This is a system procedure that moves you back to regular text screen.

4. If you want to switch back to graphmode use GRAFMODE. This procedure move you back to the graphics screen.

CREATING DRAWINGS

The following is a list of intrinsic procedures (Within TURTLEGRAPHICS) that allow you to create pictures:

PENCOLOR(color); sets color of line drawn a "turtle" moves. "Color" can be: none, white1, white2, black1, black2, green, violet, orange, blue. NOTE:

INITTURTLE sets PENCOLOR initially to NONE.

FILLSCREEN(color); colors the entire screen. Use same colors as above. If

REVERSE is used, a negative of the display is created.

MOVING COMMANDS

Vertical coordinate 0 to 191, Horizontal coordinate 0 to 279. NOTE:center of the screen is 140,95.

MOVETO(x,y) moves turtle to the pont (x,y) from current location. (x,y) must be integral lumbers).

MOVE(distance); moves turtle in the direction it is already pointing "distance" units.

TURN(angle); rotates turtle from its current angle to a new angle by "angle" degrees counterclockwise. (negative values for clockwise turn)

TURNTO(angle); rotates the turtle to "angle" degrees.

NOTE: Both "angle" and "distance" must be integers.

LABELING COMMANDS

WRITE commands are only visible on text screen (A prompt will never be seen).

WCHAR(c); displays a single charaster (CHAR) c on graph screen.

WSTRING(string); displays the "string" on the grapl screen at the current position on the turtle. (The characters are fixed size (laree) and are placed horizontally).

To write out an integar:

```
Procedure Printinteger;
var num: integer;
   charnum: string:
Begin
   Readln(num);
   STR(num, charnum);
```

½STR is a built-in procedure which con verts the integer NUM to a character equating CHARNUM. i.e. 32 becomes '32'¼

```
   Wstring(charnum)
End.
```

POSITION LOCATORS

TURTLEX - pascal intrinsic function which gives x-coordinate value of turtle's present position.

TURTLEY - same as above but for y-coordinate.

TURTLEANG- turtle's current angular direction

VIEWPORT(x1,x2,y1,y2) creates a smaller graphics screen. WARNING Set viewport to the whole screen with viewport(0,279,. Do not draw outside viewport.

UTILITY DISK #2

Now for the first time you can put both DOS 3.3 and ProDOS on the same disk! You can partition your disks to fit your needs. Utility Disk #2 will let you flip between DOS 3.3 and ProDOS without re-booting. You can also upgrade to a newer or special DOS. You can Format and Verify blocks or sectors depending on the DOS that you select. ZAP 80 will let you do block/sector editing for either DOS or ProDOS (needs an Apple //e 80 column card). The disk is unlocked and copyable and all programs will work under either DOS.

only **$ 29.95**

Special offer

Order UTILITY DISK #2 from:

PROTREE
P.O.Box 1106
Westminster, CO 80030

$ 19.95	Cost
1.50	Shipping
21.45	Total

PROTREE

LIBRARIAN

April was an extremely slow month for Apple Pi all around, the library included. Fortunately, we did get a couple contributions to the library, and you will be seeing those on the May Disk-of-the-Month.

Now is as good of a time as ever to clear up one of the most asked questions about our library: if you copy a disk from our library, can you make copies and distribute them? The answer is YES. We encourage you to distribute copies of all the disks in the library, just as long as you leave the "Denver Apple Pi" header program on the disk.

Once again, I encourage each and every one of you to write a program and contribute it to the library...remember...Public Domain Software...It only works when YOU do!

-- Trent Hein, Librarian

5 LINE CONTEST

Here is the moment that some of you have been waiting for!!! The winners of the spring Seed 5 line programming contest. This has been an interesting contest in that we had no two programs with the same idea behind them. The winning programs will be on the next Apple Pi DOM but if you are eager to try them out the code listed below. This contest was a hard one to judge. The three winners did an outstanding job and should all get a hearty appluse.

FIRST PLACE:

*** Thornton High School Computer Club ***

Program Name: MINI MAILING LIST

```
1  DIM N$(100),A$(100),C$(100),S$(100),Z$(100):D$ =  CHR$
     (13) +  CHR$ (4): HOME : PRINT  TAB( 12)"MINI MAI
     LING LIST": PRINT : PRINT "CHOOSE ONE:" TAB( 20)"
     1- CREATE FILE": PRINT  TAB( 20)"2- PRINT FILE";:
      HTAB 35: GET M: ON M GOTO 2,3: END
2 N = 1: HOME : PRINT "ENTER MAILING LIST": PRINT : GOSUB
     4: PRINT D$;"OPENMML": PRINT D$;"WRITEMML": PRINT
     N: FOR I = 1 TO N: PRINT N$(I): PRINT A$(I): PRINT
     C$(I): PRINT S$(I): PRINT Z$(I): NEXT I: PRINT D$
     ;"CLOSE": END
3  PRINT : PRINT D$;"OPENMML": PRINT D$;"READMML": INPUT
     N: FOR I = 1 TO N: INPUT N$(I),A$(I),C$(I),S$(I),
     Z$(I): NEXT I: PRINT D$;"CLOSE": PR# 1: FOR I = 1
      TO N: PRINT N$(I): PRINT A$(I): PRINT C$(I)", "S
     $(I): PRINT  TAB( 20)Z$(I): NEXT I: END
4  INPUT "NAME: ";N$(N): INPUT "STREET ADDRESS: ";A$(N
     ): INPUT "CITY: ";C$(N): INPUT "STATE: ";S$(N): INPUT
     "ZIP CODE: ";Z$(N): IF N$(N) <  > "DONE" THEN N =
     N + 1: GOTO 4
5  RETURN
```

SECOND PLACE:

*** ????? We have no name ????? ***

Program Name: SUPER SINE WAVES

```
10  TEXT : HOME : INPUT "NUMBER OF SINE WAVES TO BE MI
     XED (UP TO 10): ";NW: FOR I = 1 TO NW: PRINT " WA
     VE ";I: INPUT "AMPLITUDE: ";A(I): INPUT "WAVE LEN
     GTH: ";W(I): NEXT : PRINT "WHEN WAVE IS DONE HIT
     ESC TO QUIT, ANYTHING ELSE TO CONT.": GET CN$
20  HGR2 : FOR X = SCR * 280 TO SCR * 280 + 279:Y = 0:
      FOR I = 1 TO NW:Y = Y + ( SIN (X / W(I)) * A(I))
     : NEXT I:Y = Y + 96: HPLOT X - (SCR * 280),96: IF
     Y < 0 THEN Y = 0
30  IF Y > 191 THEN Y = 191
40  HPLOT X - (SCR * 280),Y: NEXT X: GET ANS$: IF ANS$
      <  > CHR$ (27) THEN SCR = SCR + 1: GOTO 20
```

THIRD PLACE:

*** Randy Herman ***

Program Name: MEMORY RANGE FINDER

```
10  HOME : HTAB 12: PRINT "--- BYTE SEARCH ---
     ": PRINT : INPUT "STARTING ADDRESS...";S:
      INPUT "ENDING ADDRESS.....";E: PRINT "SH
     OW HI BIT SET Y/N ";: GET HB$: PRINT : PRINT
     : FOR I = 1 TO 39: PRINT "=";: NEXT : PRINT
     : PRINT : POKE 34,8
20  LET N = E - S: FOR I = 0 TO N:P = I + S: PRINT
     P;" = ";: IF  PEEK (P) > 127 AND HB$ = "Y
     " THEN  INVERSE
30  PRINT  PEEK (P): NORMAL : IF  PEEK (37) =
     20 THEN  VTAB 22: PRINT "PRESS KEY TO CON
     TINUE";: GET Z$: HOME
40  NEXT I: VTAB 22: PRINT "PRESS ANY KEY TO R
     E-RUN ";: GET Z$: POKE 34,0: GOTO 10
```

NOTE:

A few quick notes are in order. The Mail list program needs to be RUN for each operation. If you are making a file or printing a file the operation will exit when done to Applesoft. To do another operation you must RUN the program again. The Super Sine Waves program is a HIRES program and will print on a color screen (if you have one). You must set the color yourself with a line like this.

```
5 HCOLOR= x   : REM x = the color code
```

APPLE PI NEEDS YOU!

IT'S THAT TIME OF YEAR WHEN APPLE PI CHANGES OFFICERS. THE BOARD MEMBERS ARE VOTED INTO OFFICE BUT ALL OTHER OFFICERS VOLUNTEER. HOW ABOUT YOU?

SALES OFFICER / SOFTWARE LIBRARIAN / PUBLICATIONS LIBRARIAN / NEW MEMBER CONTACT NEWSLETTER / SIGS / IAC REPRESENTATIVE BBS SYSOP / STORE REPRESENTATIVE /

---

denver apple pi Seed

P.O. Box 17467
Denver, CO 80217

BULK RATE
U.S. POSTAGE
PAID
DENVER, COLO.
PERMIT NO. 3182

Price $1.50